Clustering algorithms

Konstantinos Koutroumbas

Unit 10

– Graph theory-based clustering algorithms
– Competitive learning clustering algorithms
– Valley seeking clustering algorithms
– Branch & bound clustering algorithms
– Simulated annealing-based clustering algorithm

koutroum@noa.gr

Other clustering algorithms

- The following types of algorithms will be considered:
  - Graph theory based clustering algorithms.
  - Competitive learning algorithms.
  - Valley seeking clustering algorithms.
  - Cost optimization clustering algorithms based on:
    - Branch and bound approach.
    - Simulated annealing methodology.
    - Deterministic annealing.
    - Genetic algorithms.
  - Density-based clustering algorithms.
  - Clustering algorithms for high dimensional data sets.

Graph theory based clustering algorithms

In principle, such algorithms are capable of detecting clusters of various shapes, at least when they are well separated.

In the sequel we discuss algorithms that are based on:

- The Minimum Spanning Tree (MST).
- Regions of influence.
- Directed trees.

Graph theory based clustering algorithms

Minimum Spanning Tree (MST) algorithms

Preliminaries: Let

- G be the complete graph, each node of which corresponds to a point of the data set X.

- $e = (\boldsymbol{x}_i, \boldsymbol{x}_j)$ denote an edge of G connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

- $w_e \equiv d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ denote the weight of the edge e.

Definitions:

- Two edges e_1 and e_2 are k steps away from each other if the minimum path that connects a vertex of e_1 and a vertex of e_2 contains $k - 1$ edges.

- A Spanning Tree of G is a connected graph that:
  - Contains all the vertices of the graph.
  - Has no loops.

- The weight of a Spanning Tree is the sum of weights of its edges.

- A Minimum Spanning Tree (MST) of G is a spanning tree with minimum weight (when all w_e's are different from each other, the MST is unique).

Graph theory based clustering algorithms

Minimum Spanning Tree (MST) algorithms (cont)

Sketch of the algorithm:

- Determine the MST of G.
- Remove the edges that are "unusually" large compared with their neighboring edges (inconsistent edges).
- Identify as clusters the connected components of the MST, after the removal of the inconsistent edges.

Identification of inconsistent edges.

For a given edge e of the MST of G:

- Consider all the edges (except e) that lie k steps away (at the most) from e.
- Determine the mean m_e and the standard deviation σ_e of their weights.
- If w_e lies more than q (typically $q = 2$) standard deviations σ_e away from m_e, then:
  - e is characterized as inconsistent.
- Else
  - e is characterized as consistent.
- End if

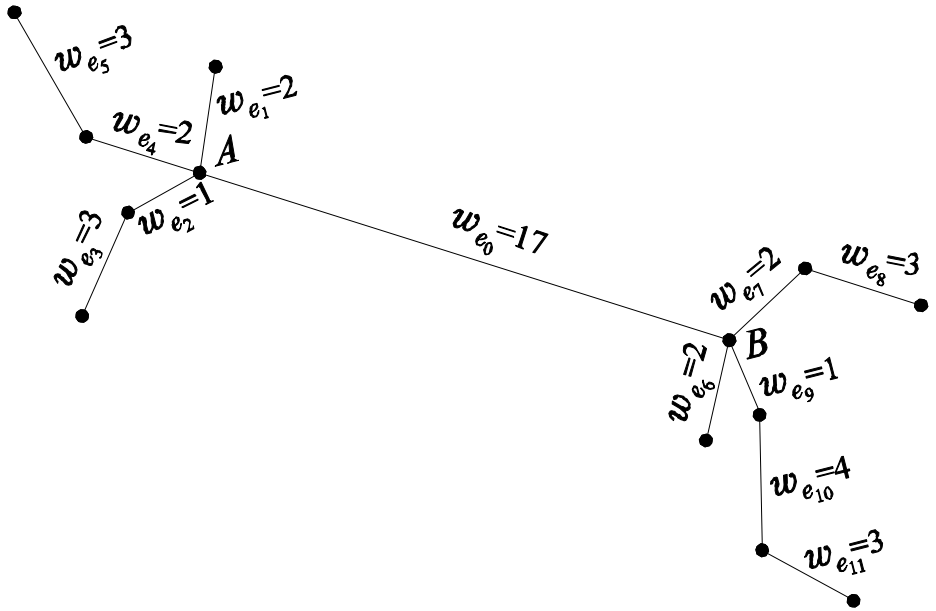
Graph theory based clustering algorithms

Minimum Spanning Tree (MST) algorithms (cont)

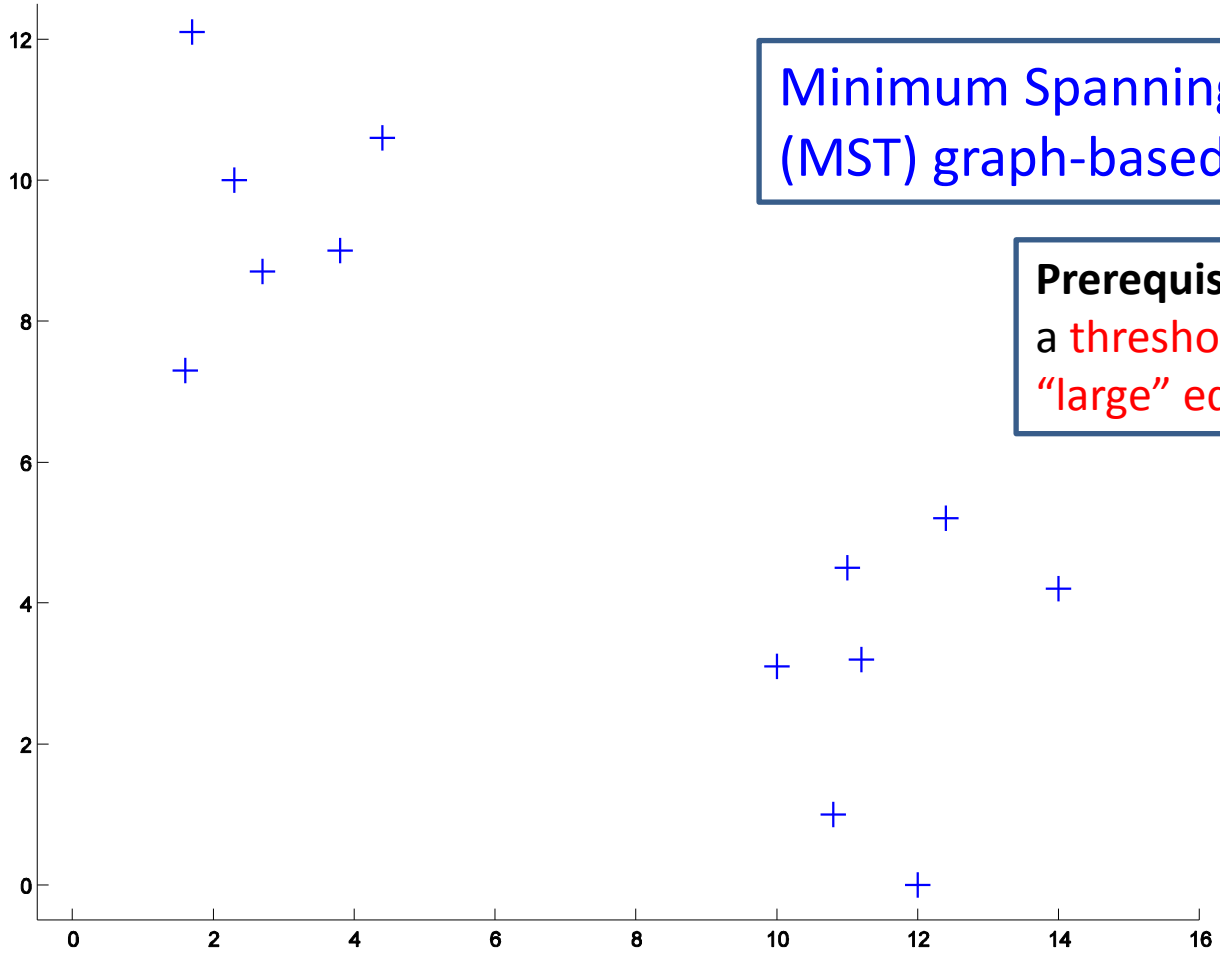
Example:

- For the MST in the figure and for $k = 2$ and $q = 3$ we have:
- For e_0: $w_{e_0} = 17, m_{e_0} = 2.3, \sigma_{e_0} = 0.95$. w_{e_0} lies 15.5 standard deviations σ_{e_0} away from m_{e_0}, hence it is inconsistent.

- For e_{11}: $w_{e_{11}} = 3, m_{e_{11}} = 2.5, \sigma_{e_{11}} = 2.12$. $w_{e_{11}}$ lies 0.24 standard deviations $\sigma_{e_{11}}$ away from $m_{e_{11}}$, hence it is consistent.



Graph theory based clustering algorithms

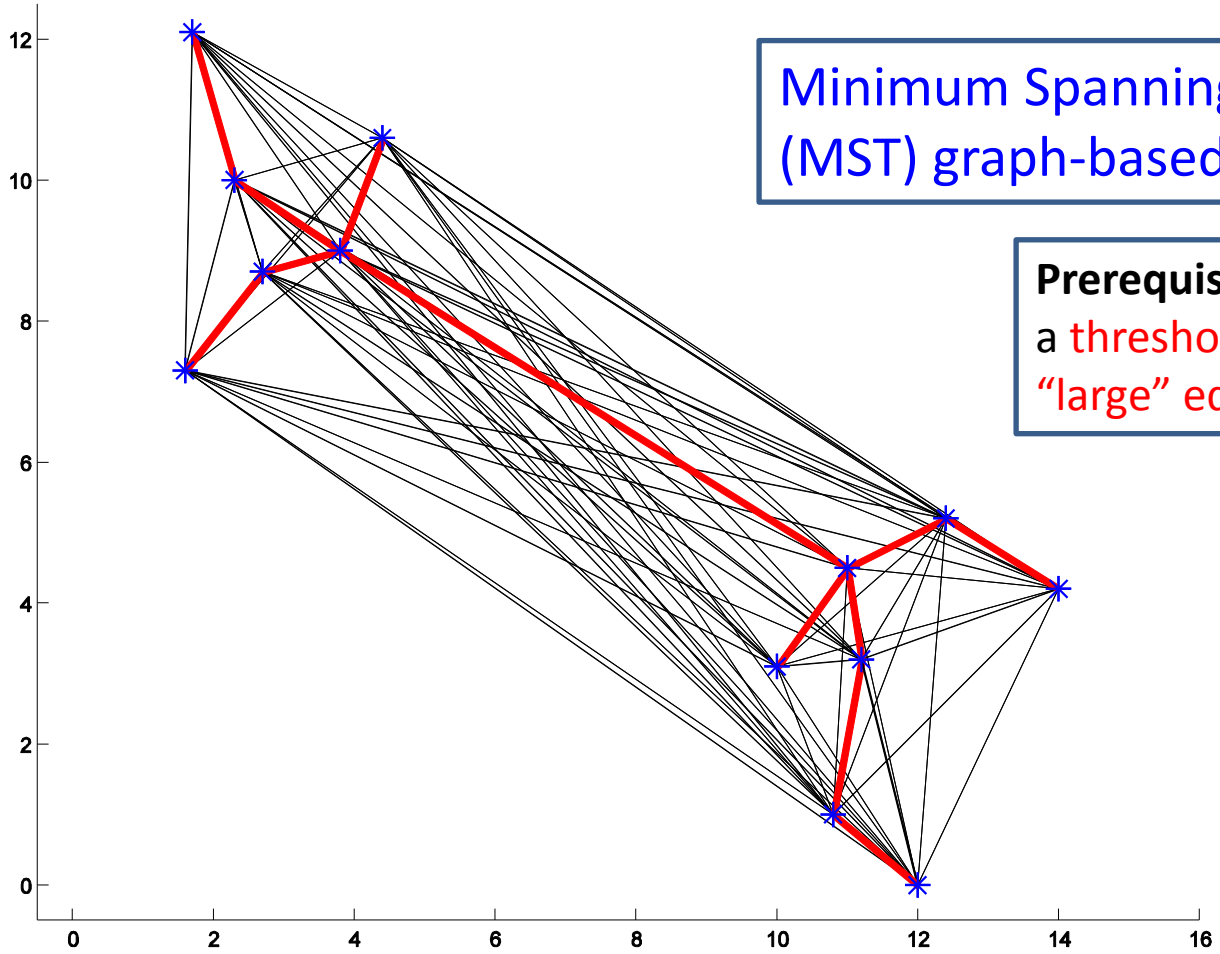


Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms

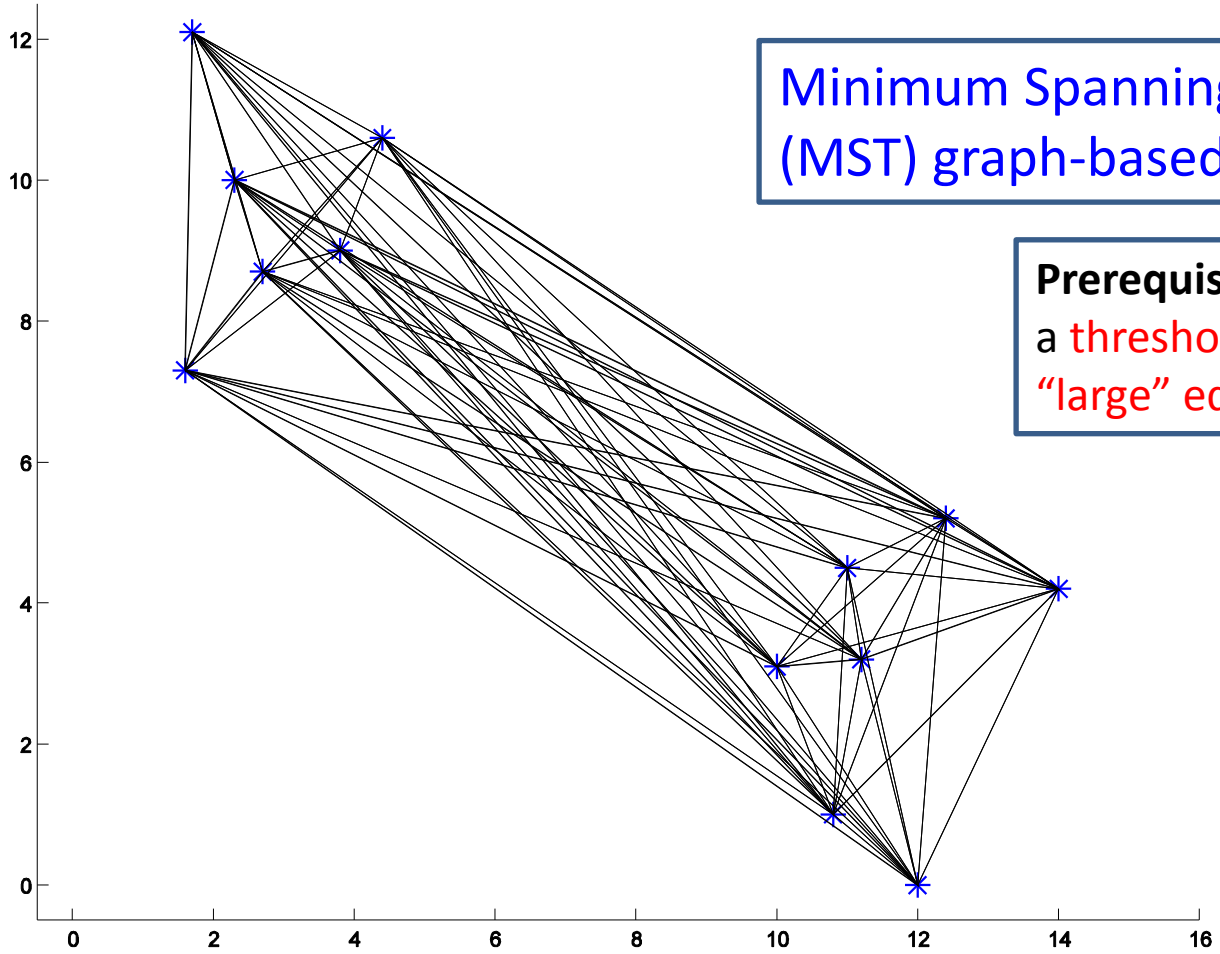


Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms

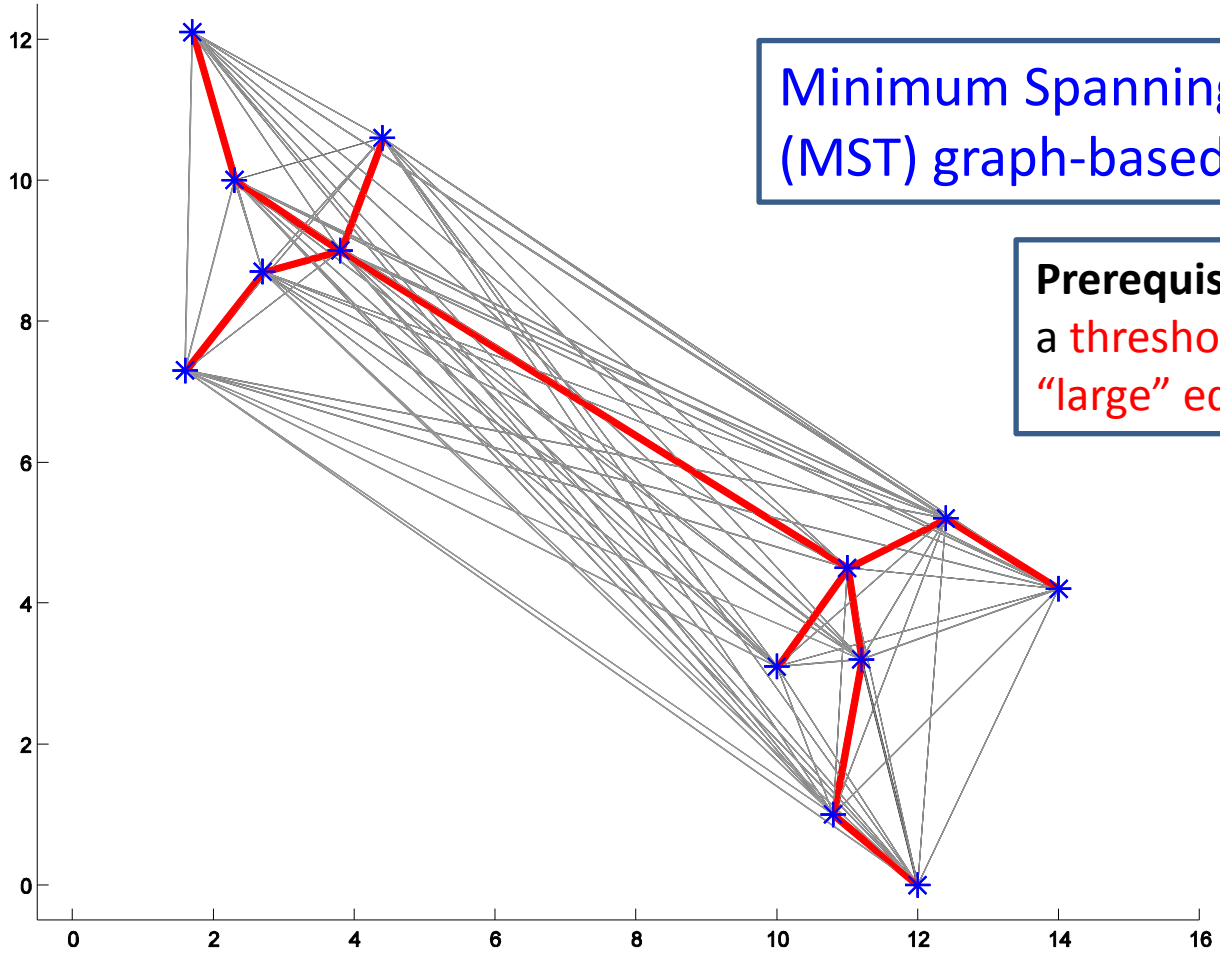


Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms

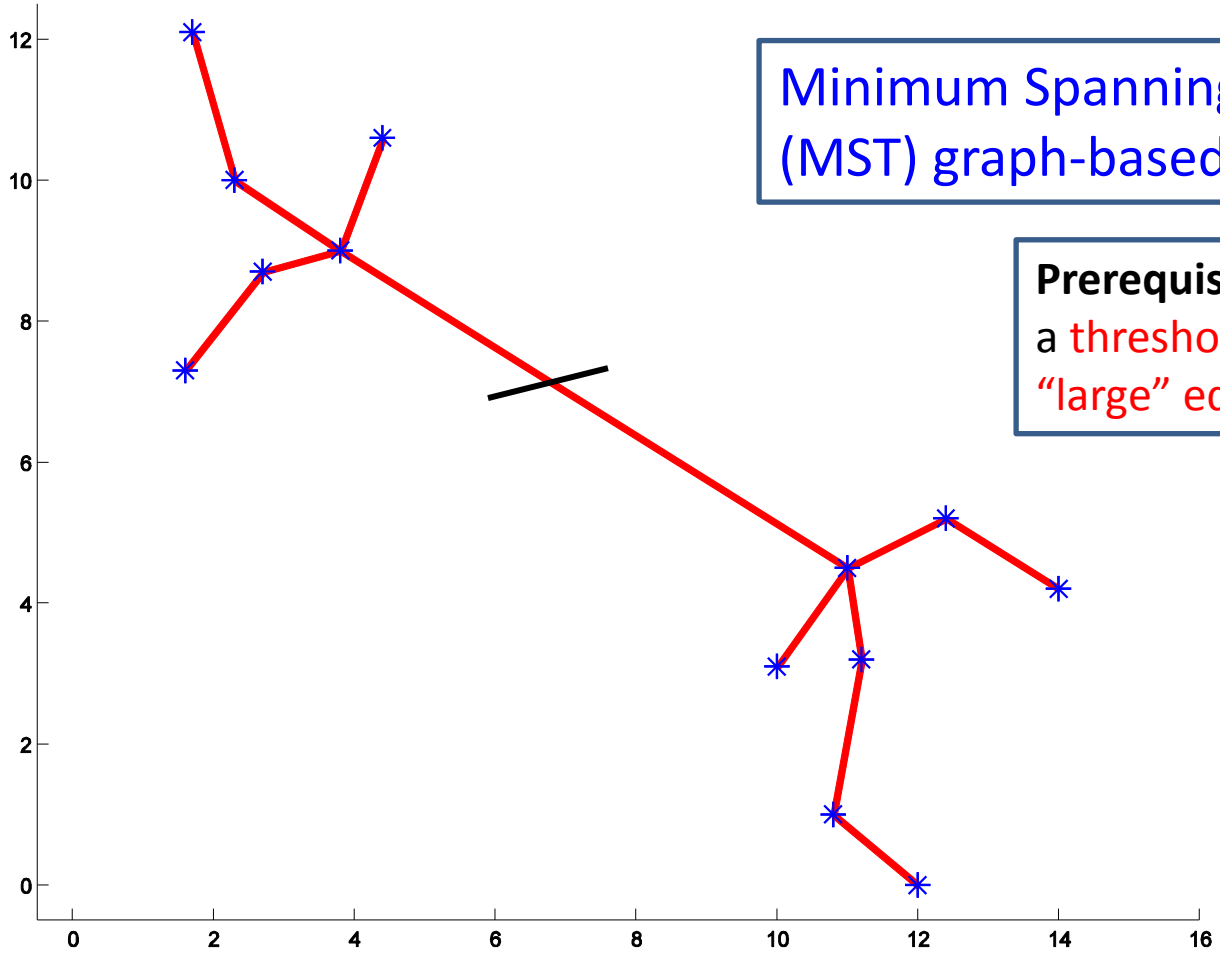


Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms

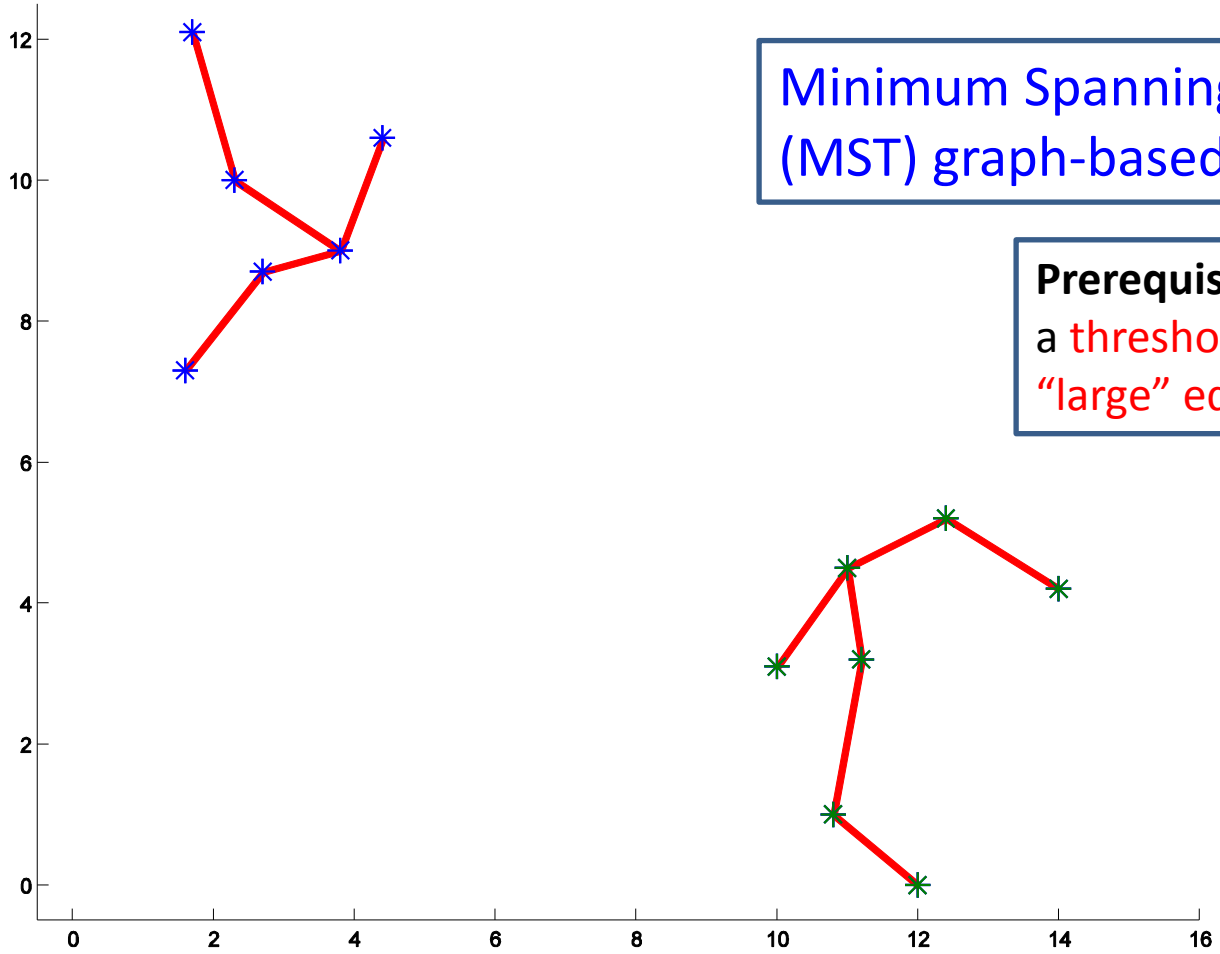


Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms



Minimum Spanning Tree (MST) graph-based algorithm

Prerequisite: Definition of a threshold for identifying "large" edges.

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph **and cut** the **"unusually large"** edges.
- The remaining sub-graphs **correspond** to the clusters.

Graph theory based clustering algorithms

Minimum Spanning Tree (MST) algorithms (cont)

Remarks:

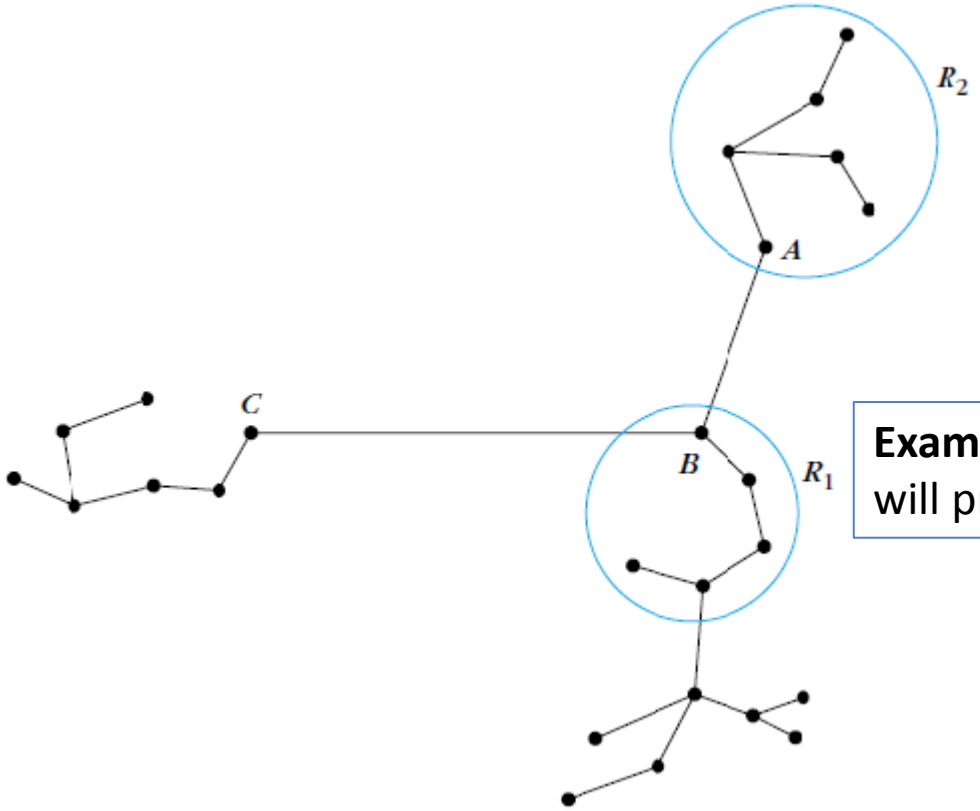
- The algorithm **depends** on the choices of k and q.
- The algorithm is insensitive to the order of consideration of the data points.
- No initial conditions are required, no convergence issues are arised.
- The algorithm works well for many cases where the clusters are well separated.

Graph theory based clustering algorithms

Minimum Spanning Tree (MST) algorithms (cont)

Remarks:

- A **problem** may occur when a "large" edge e has another "large" edge as its neighbor. In this case, e is likely not to be characterized as inconsistent and the algorithm may fail to unravel the underlying clustering structure correctly.



Example: The vectors of the regions R_1 and R_2 will probably be assigned to the same cluster.

Graph theory based clustering algorithms

Algorithms based on Regions of Influence (ROI)

Definition: The region of influence of two distinct vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$ is defined as:

$$R(\boldsymbol{x}_i, \boldsymbol{x}_j) = \{\boldsymbol{x}:\ cond(d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j), d(\boldsymbol{x}_i, \boldsymbol{x}_j)), \boldsymbol{x}_i \neq \boldsymbol{x}_j\}$$

where $cond(d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j), d(\boldsymbol{x}_i, \boldsymbol{x}_j))$ may be defined as:

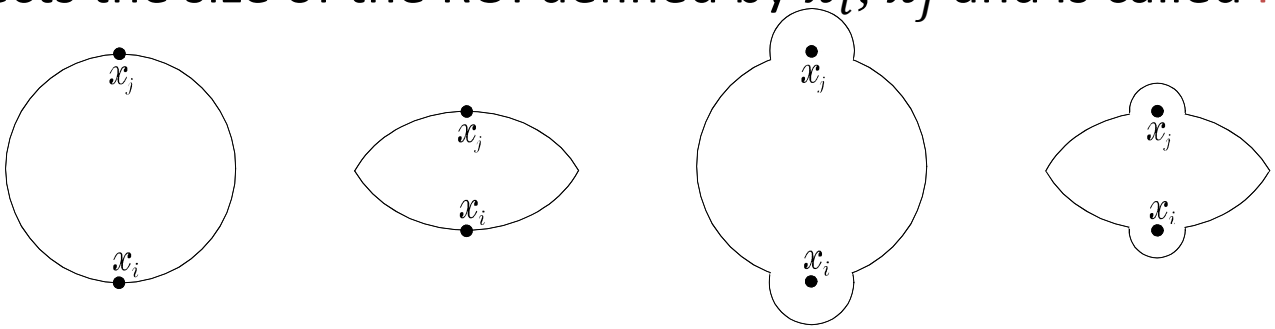
a) $d^2(\boldsymbol{x}, \boldsymbol{x}_i) + d^2(\boldsymbol{x}, \boldsymbol{x}_j) < d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)$,

b) $\max\{d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j)\} < d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$,

c) $\left(d^2(\boldsymbol{x}, \boldsymbol{x}_i) + d^2(\boldsymbol{x}, \boldsymbol{x}_j) < d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) OR \left(\sigma \min\{d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j)\} < d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)$,

d) $\left(\max\{d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j)\} < d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}\right) OR \left(\sigma \min\{d(\boldsymbol{x}, \boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_j)\} < d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)$

where σ affects the size of the ROI defined by $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and is called relative edge consistency.



(a) (b) (c) (d)

Graph theory based clustering algorithms

Algorithms based on Regions of Influence (cont)

Algorithm based on ROI

- For $i = 1$ to N
  - For $j = i + 1$ to N
    - **Determine** the region of influence $R(\boldsymbol{x}_i, \boldsymbol{x}_j)$
    - If $R(\boldsymbol{x}_i, \boldsymbol{x}_j) \cap \left(X - \{\boldsymbol{x}_i, \boldsymbol{x}_j\}\right) = \emptyset$ then
      - Add the edge connecting $\boldsymbol{x}_i, \boldsymbol{x}_j$.
    - End if
  - End For
- End For

Determine the connected components of the resulted graph and **identify** them as clusters.

In words:

- The edge $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is **added** to the graph **if** no other $\boldsymbol{x}_q \in X$ lies in $R(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

- Since for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ close to each other it is likely that $R(\boldsymbol{x}_i, \boldsymbol{x}_j)$ contains no other vectors in X, it is expected that close to each other points will be assigned to the same cluster.

Graph theory based clustering algorithms

Algorithms based on Regions of Influence (cont)

Remarks:

- The algorithm is insensitive to the order in which the pairs are considered.
- In order to exclude (possible) edges connecting distant points, one could use a procedure like the one described previously for removing "unusually large" edges.
- In the choices of $cond$ in (c) and (d), σ must be chosen *a priori*.
- For the resulting graphs:
  - if the choice (a) is used for $cond$, they are called relative neighborhood graphs (RNGs)
  - if the choice (b) is used for $cond$, they are called Gabriel graphs (GGs)
- Experimental results show that better clusterings are produced when (c) and (d) conditions are used in the place of $cond$, instead of (a) and (b).

Graph theory based clustering algorithms

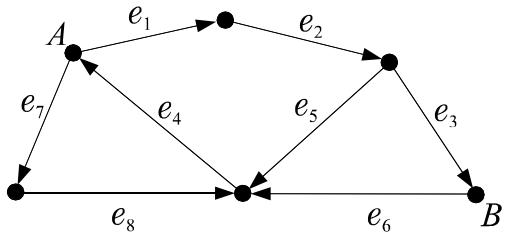
Algorithms based on Directed Trees

Definitions:

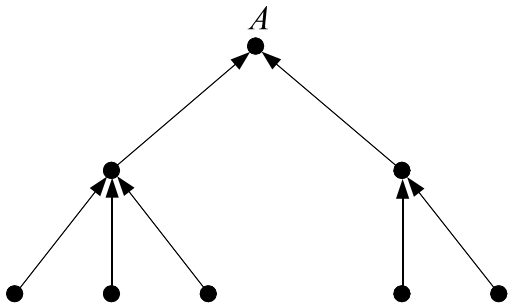
- A directed graph is a graph whose edges are directed.

- A set of edges $e_{i_1}, \dots, e_{i_q}$ **constitute** a directed path from a vertex A to a vertex B, if,
  - A is the initial vertex of e_{i_1}
  - B is the final vertex of e_{i_q}
  - The destination vertex of the edge $e_{i_j}, j = 1, \dots, q-1$, **is** the departure vertex of the edge $e_{i_{j+1}}$.

  (In figure (a) the sequence e_1, e_2, e_3 constitute a directed path connecting the vertices A and B).



(a)



(b)

Graph theory based clustering algorithms

Algorithms based on Directed Trees (cont)

- A directed tree is a directed graph with a specific node A, known as root, such that,
  - From every node $B \neq A$ of the tree **departs** exactly one edge.
  - No edge **departs** from A.
  - No circles are encountered (see figure (b) in the previous slide).

- The neighborhood of a point $\boldsymbol{x}_i \in X$ is defined as

$$\rho_i(\theta) = \{\boldsymbol{x}_j \in X : d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \theta, \boldsymbol{x}_i \neq \boldsymbol{x}_j\}$$

  where θ determines the neighborhood size.

- Also let
  - $n_i = |\rho_i(\theta)|$ be the number of points of X lying within $\rho_i(\theta)$
  - $g_{ij} = (n_j - n_i)/d(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Main philosophy of the algorithm

Identify the directed trees in a graph whose vertices are points of X, so that each directed tree corresponds to a cluster.

Graph theory based clustering algorithms

Algorithms based on Directed Trees (cont.)

Clustering Algorithm based on Directed Trees

$g_{ij} = (n_j - n_i)/d(\boldsymbol{x}_i, \boldsymbol{x}_j)$

- **Set** θ to a specific value.
- **Determine** n_i, $i = 1, \ldots, N$.
- **Compute** g_{ij}, $i, j = 1, \ldots, N$, $i \neq j$.
- For $i = 1$ to N
  - If $n_i = 0$ then
    - $\boldsymbol{x}_i$ is the root of a new directed tree.
  - Else
    - Determine $\boldsymbol{x}_r$ such that $g_{ir} = max_{x_j \in \rho_i(\theta)} g_{ij}$
    - If $g_{ir} < 0$ then
      - $\boldsymbol{x}_i$ is the root of a new directed tree.
    - Else if $g_{ir} > 0$ then
      - $\boldsymbol{x}_r$ is the parent of $\boldsymbol{x}_i$ (there exists a directed edge from $\boldsymbol{x}_i$ to $\boldsymbol{x}_r$).

Graph theory based clustering algorithms

Algorithms based on Directed Trees (cont.)

Clustering Algorithm based on Directed Trees

- Else if $g_{ir} = 0$ then
  - Define $T_i = \{\boldsymbol{x}_j : \boldsymbol{x}_j \in \rho_i(\theta), g_{ij} = 0\}$.
  - Eliminate all the elements $\boldsymbol{x}_j \in T_i$, for which there exists a directed path from $\boldsymbol{x}_j$ to $\boldsymbol{x}_i$.
  - If the resulting T_i is empty then
    - $\boldsymbol{x}_i$ is the root of a new directed tree
  - Else
    - The parent of $\boldsymbol{x}_i$ is $\boldsymbol{x}_q$ such that $d(\boldsymbol{x}_i, \boldsymbol{x}_q) = min_{\boldsymbol{x}_s \in T_i} d(\boldsymbol{x}_i, \boldsymbol{x}_s)$.
  - End if
- End if
- End if

- End for
- **Identify** as clusters the directed trees formed above.

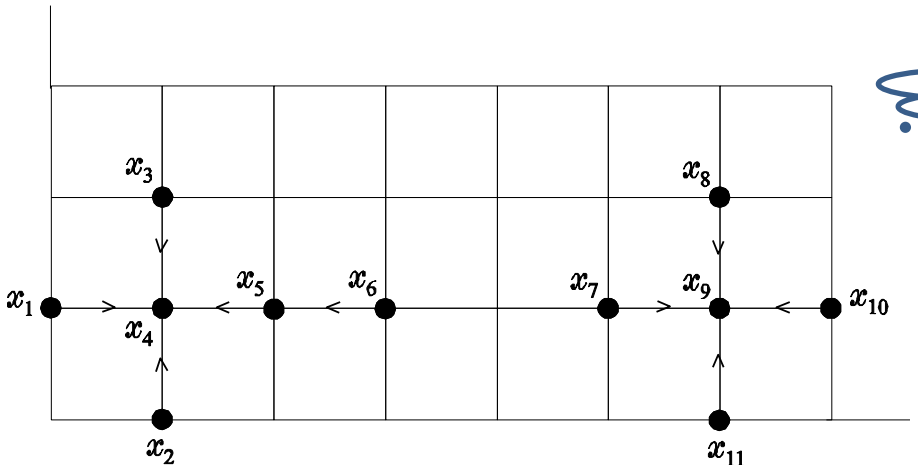
Graph theory based clustering algorithms

Algorithms based on Directed Trees (cont.)

Remarks:

- The root $\boldsymbol{x}_i$ of a directed tree is the point in $\rho_i(\theta)$ with the most dense neighborhood.
- The branch that handles the case $g_{ir} = 0$ **ensures** that no circles occur.
- The algorithm is sensitive to the order of consideration of the data points.
- For proper choice of θ and large N, this scheme **behaves** as a mode-seeking algorithm (see below).

Example: In the figure below, the size of the edge of the grid is 1 and $\theta = 1.1$. The above algorithm gives the directed trees shown in the figure.



$$g_{ij} = (n_j - n_i)/d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Competitive learning clustering algorithms

The main idea

- **Employ** a set of representatives $\boldsymbol{w}_j$ (in the sequel we consider only point representatives).
- **Move** them to regions of the vector space that are "dense" in vectors of X.

Comments

- In general, representatives are **updated** each time a new vector $\boldsymbol{x} \in X$ is presented to the algorithm (pattern mode algorithms).
- These algorithms do not necessarily stem from the optimization of a cost function.

The strategy

- For a given vector $\boldsymbol{x}$
  - All representatives **compete** to each other
  - The winner (representative that lies closest to $\boldsymbol{x}$) **moves** towards $\boldsymbol{x}$.
  - The losers (the rest of the representatives) either remain unchanged **or** they move towards $\boldsymbol{x}$ but at a much slower rate.

Competitive learning clustering algorithms

Generalized Competitive Learning Scheme (GCLS)

$t = 0$

$m = m_{init}$ (initial number of representatives)

(A) **Initialize** any other necessary parameters (depending on the specific algorithm).

Repeat

- $t = t + 1$
- **Present** a new randomly selected $\boldsymbol{x} \in X$ to the algorithm.
- (B) **Determine** the winning representative $\boldsymbol{w}_j(t-1)$.
- (C) *If* ((***x*** *is* ***not "similar"*** *to* $\boldsymbol{w}_j(t-1)$) OR (*other condition*)) AND $(m < m_{max})$ *then*
  - $m = m + 1$
  - $\boldsymbol{w}_m = \boldsymbol{x}$

  Else
  - (D) *Parameter updating*

$$\boldsymbol{w}_q(t) = \begin{cases} \boldsymbol{w}_q(t-1) + \eta h\left(\boldsymbol{x}, \boldsymbol{w}_q(t-1)\right), & if\ \boldsymbol{w}_q \equiv \boldsymbol{w}_j\ (winner) \\ \boldsymbol{w}_q(t-1) + \eta' h\left(\boldsymbol{x}, \boldsymbol{w}_q(t-1)\right), & otherwise \end{cases}$$

End

(E) **Until** (convergence occurred) OR $(t > t_{max})$

Assign each $\boldsymbol{x} \in X$ to the cluster whose representative $\boldsymbol{w}_j$ lies closest to $\boldsymbol{x}$.

maximum allowable number of clusters

maximum allowable number of iterations

Competitive learning clustering algorithms

Remarks:

- $h(\boldsymbol{x}, \boldsymbol{w}_q)$ is an appropriately defined function (see below).

- η and η' are the learning rates controlling the updating of the winner and the losers, respectively (η' may differ from looser to looser).

- A threshold of similarity Θ (carefully chosen) controls the similarity between $\boldsymbol{x}$ and its closest representative $\boldsymbol{w}_j$.
  - If $d(\boldsymbol{x}, \boldsymbol{w}_j) > \Theta$, for some distance measure, $\boldsymbol{x}$ and $\boldsymbol{w}_j$ are considered as dissimilar.

- A termination criterion may be the small variation of $\boldsymbol{W} = [\boldsymbol{w}_1^T, \ldots, \boldsymbol{w}_m^T]^T$ for at least N iterations (N is the cardinality of X), i.e., for any pair of t_1, t_2, with $(p-1) \cdot N \leq t_1, t_2 \leq p \cdot N, p \in Z$, to hold $||\boldsymbol{W}(t_1) - \boldsymbol{W}(t_2)|| < \varepsilon$.

- With appropriate choices of (A), (B), (C) and (D), most competitive learning algorithms may be viewed as special cases of GCLS.

Competitive learning clustering algorithms

Basic Competitive Learning Algorithm

Here the number of representatives m is **constant**.

The algorithm

- $t = 0$
- **Repeat**
  - $t = t + 1$
  - **Present** a new randomly selected $\boldsymbol{x} \in X$ to the algorithm.
  - (B) **Determine** the winning representative $\boldsymbol{w}_j$ on $\boldsymbol{x}$ as the one for which

$$d(\boldsymbol{x}, \boldsymbol{w}_j(t-1)) = min_{k=1,\ldots,m} d(\boldsymbol{x}, \boldsymbol{w}_k(t-1)) \ (*).$$

  - (D) *Parameter updating*

$\eta \in (0,1)$

$$\boldsymbol{w}_q(t) = \begin{cases} \boldsymbol{w}_q(t-1) + \eta\left(\boldsymbol{x} - \boldsymbol{w}_q(t-1)\right), & if\, \boldsymbol{w}_q \equiv \boldsymbol{w}_j \ (winner) \\ \boldsymbol{w}_q(t-1), & otherwise \end{cases}$$

  - End
- (E) **Until** (convergence occurred) OR $(t > t_{max})$
- **Assign** each $\boldsymbol{x} \in X$ to the cluster whose representative $\boldsymbol{w}_j$ lies closest to $\boldsymbol{x}$.

(*) $d(\cdot)$ may be any distance (e.g., Euclidean dist., Itakura-Saito distortion).
Also, similarity measures **may** be **used** (in this case min is **replaced by** max).

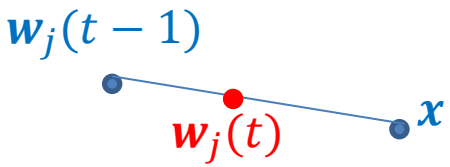
Competitive learning clustering algorithms

Basic Competitive Learning Algorithm (cont.)

Remarks:

- In this scheme losers **remain** unchanged. The winner, after the updating, **lies** in the line segment formed by $\boldsymbol{w}_j\ (t-1)$ and $\boldsymbol{x}$.

$$\boldsymbol{w}_j(t) = \boldsymbol{w}_j(t-1) + \eta\left(\boldsymbol{x} - \boldsymbol{w}_j(t-1)\right)$$
$$\Leftrightarrow \boldsymbol{w}_j(t) = (1-\eta)\boldsymbol{w}_j(t-1) + \eta\boldsymbol{x}$$



- *A priori* knowledge of the number of clusters m is required.

- If a representative is initialized far away from the regions where the points of X lie, it will never win.
  Possible solution: **Initialize** all representatives using vectors of X.

- Versions of the algorithm with variable learning rate have also been studied. Specifically, $\eta_t \to 0$, as $t \to \infty$, but not too fast(*)

(*) $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} {\eta_t}^2 < \infty$ (**stochastic** algorithms)

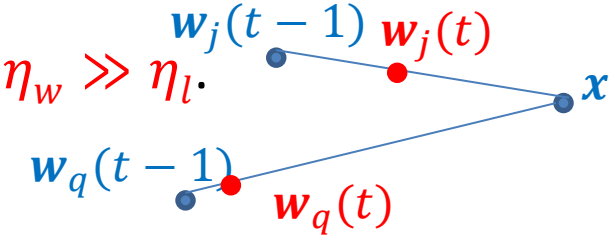
Competitive learning clustering algorithms

Leaky Learning Algorithm

The same with the Basic Competitive Learning Algorithm except part (D), the updating equation of the representatives, which becomes

$$\boldsymbol{w}_q(t) = \begin{cases} \boldsymbol{w}_q(t-1) + \eta_w h\left(\boldsymbol{x} - \boldsymbol{w}_q(t-1)\right), & if\ \boldsymbol{w}_q \equiv \boldsymbol{w}_j\ (winner) \\ \boldsymbol{w}_q(t-1) + \eta_l h\left(\boldsymbol{x} - \boldsymbol{w}_q(t-1)\right), & otherwise \end{cases}$$

where η_w and η_l are the learning rates in $(0, 1)$ and $\eta_w \gg \eta_l$.



Remarks:

- All representatives **move** towards $\boldsymbol{x}$ but the losers move at a much slower rate than the winner does.
- The algorithm does not suffer from the problem of poor initialization of the representatives (why?).
- An algorithm in the same spirit is the "neural-gas" algorithm, where η_l varies from loser to loser and decays as the corresponding representatives lie away from $\boldsymbol{x}$. This algorithm **results** from the optimization of a cost function.

Competitive learning clustering algorithms

Conscientious Competitive Learning Algorithms

Main Idea: **Discourage** a representative $\boldsymbol{w}_q$ from winning *if it has won many times in the past*. Do this by assigning a "conscience" to each representative.

A simple implementation

- **Equip** each representative $\boldsymbol{w}_q$, $q = 1, \ldots, m$, with a counter f_q that **counts** the times that $\boldsymbol{w}_q$ wins.
- At part (A) (initialization stage) of GCLS set $f_q = 1$, $q = 1, \ldots, m$.
- Define the distance $d^*(\boldsymbol{x}, \boldsymbol{w}_q)$ as

$$d^*(\boldsymbol{x}, \boldsymbol{w}_q) = d(\boldsymbol{x}, \boldsymbol{w}_q) f_q.$$

(the distance is penalized to discourage representatives that have won many times)

- Part (B) becomes
  - The representative $\boldsymbol{w}_j$ is the winner on $\boldsymbol{x}$ if

$$d^*(\boldsymbol{x}, \boldsymbol{w}_j) = min_{q=1,\ldots,m} d^*(\boldsymbol{x}, \boldsymbol{w}_q)$$

  - Set $f_j(t) = f_j(t-1) + 1$

- Parts (C) and (D) are the same as in the Basic Competitive Learning Algorithm
- Also $m = m_{init} = m_{max}$

Competitive learning clustering algorithms

Conscientious Competitive Learning Algorithms

The algorithm

- Set $f_q = 1, q = 1, \dots, m$
- $t = 0$
- **Repeat**
  - $t = t + 1$
  - **Present** a new randomly selected $\boldsymbol{x} \in X$ to the algorithm.
  - (B) **Compute** $d^*\big(\boldsymbol{x}, \boldsymbol{w}_q(t-1)\big) = d\big(\boldsymbol{x}, \boldsymbol{w}_q(t-1)\big) f_q \ , q = 1, \dots, m.$

    Determine the winning representative $\boldsymbol{w}_j$ on $\boldsymbol{x}$ as the one for which

    $$d^*\big(\boldsymbol{x}, \boldsymbol{w}_j(t-1)\big) = min_{q=1,\dots,m} d^*\big(\boldsymbol{x}, \boldsymbol{w}_q(t-1)\big).$$

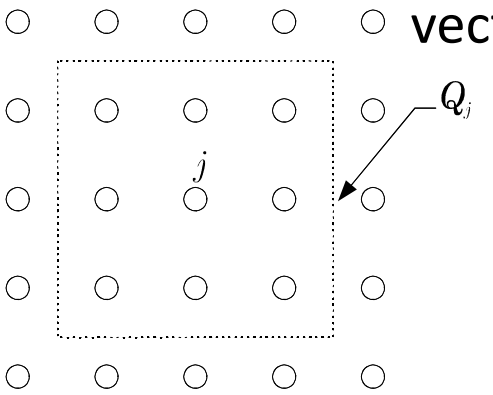
    Set $f_j\ (t) = f_j(t-1) + 1$
  - (D) *Parameter updating*

$$\boldsymbol{w}_q(t) = \begin{cases} \boldsymbol{w}_q(t-1) + \eta\left(\boldsymbol{x} - \boldsymbol{w}_q(t-1)\right), & if\ \boldsymbol{w}_q \equiv \boldsymbol{w}_j\ (winner) \\ \boldsymbol{w}_q(t-1), & otherwise \end{cases}$$

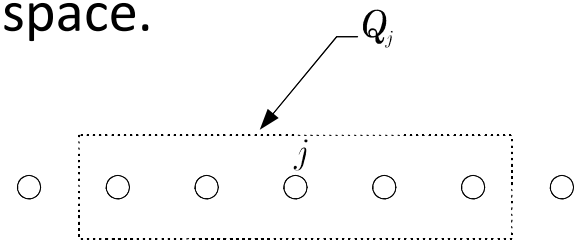
  - End
- (E) **Until** (convergence occurred) OR $(t > t_{max})$
- **Assign** each $\boldsymbol{x} \in X$ to the cluster whose representative $\boldsymbol{w}_j$ lies closest to $\boldsymbol{x}$.

Self-organizing maps (*)

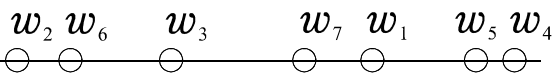
- It is **used** for data visualization (maps high dim. Data→ 1-d or 2-d maps) and ("loose") clustering.
- Here interrelation between representatives is assumed.
- For each representative $\boldsymbol{w}_j$ a topological neighborhood of representatives $Q_j(t)$ is defined, centered at $\boldsymbol{w}_j$.
- As t (no. of iterations) increases, $Q_j(t)$ shrinks and concentrates around $\boldsymbol{w}_j$.
- The neighborhood is defined with respect to the indices j and it is independent of the geometrical distances between representatives in the vector space.



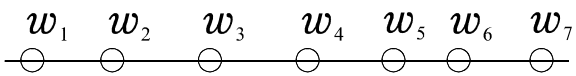
(a)



(b)



(c)



(d)

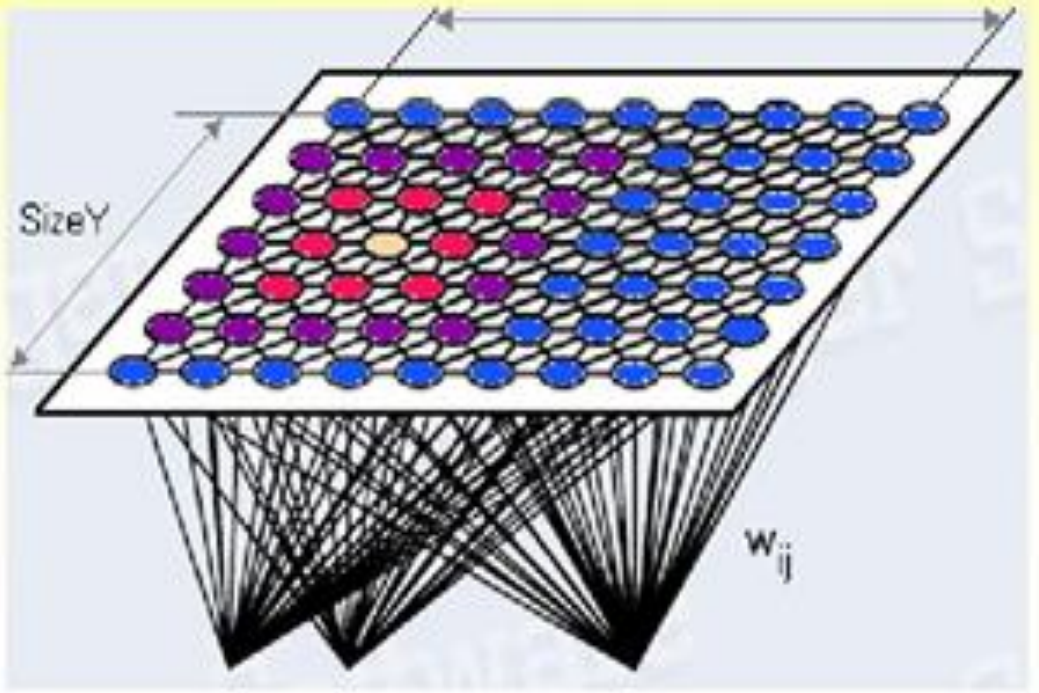
Assuming that in fig. (c) the neighborhood of $\boldsymbol{w}_j$ **constitutes** of $\boldsymbol{w}_{j-1}$ and $\boldsymbol{w}_{j+1}$,

$\boldsymbol{w}_2$ and $\boldsymbol{w}_4$ are the **topo-logical neighbors** of $\boldsymbol{w}_3$ **although**

$\boldsymbol{w}_6$ and $\boldsymbol{w}_7$ are closer in terms of the geometrical distance to $\boldsymbol{w}_3$)

Self-organizing maps (*)

- Here interrelation between representatives is assumed.
- For each representative $\boldsymbol{w}_j$ a neighborhood of representatives $Q_j(t)$ is defined, centered at $\boldsymbol{w}_j$.
- As t (number of iterations) increases, $Q_j(t)$ shrinks and concentrates around $\boldsymbol{w}_j$.
- The neighborhood is defined with respect to the indices j and it is independent of the distances between representatives in the vector space.



Self-organizing maps (*)

- If $\boldsymbol{w}_j$ wins on the current input $\boldsymbol{x}$ all the representatives in $Q_j(t)$ are updated (Self Organizing Map (SOM) scheme).
- SOM (in its simplest version) may be viewed as a special case of GCLS if

  - Parts (A), (B) and (C) are defined as in the basic competitive learning scheme.
  - In part (D), if $\boldsymbol{w}_j$ wins on $\boldsymbol{x}$, the updating equation becomes:

$$\boldsymbol{w}_k(t) = \begin{cases} \boldsymbol{w}_k(t-1) + {\eta_t}^{k,j}\big(\boldsymbol{x} - \boldsymbol{w}_k(t-1)\big), & if\ \boldsymbol{w}_k \in Q_j(t) \\ \boldsymbol{w}_k(t-1), & otherwise \end{cases}$$

  where ${\eta_t}^{k,j}$ is a *variable* learning rate, which decreases with t and with the topological distance between the k-th and the j-th representatives.

- After convergence, neighboring representatives also lie "close" in terms of their geometrical distance in the vector space (topographical ordering) (see fig. (d)).

Self-organizing maps (*)

The algorithm

- $t = 0$
- **Repeat**
  - $t = t + 1$
  - **Present** a new randomly selected $\boldsymbol{x} \in X$ to the algorithm.
  - (B) **Determine** the winning representative $\boldsymbol{w}_j$ on $\boldsymbol{x}$ as the one for which

$$d\big(\boldsymbol{x}, \boldsymbol{w}_j(t-1)\big) = \min_{k=1,\ldots,m} d(\boldsymbol{x}, \boldsymbol{w}_k(t-1))$$

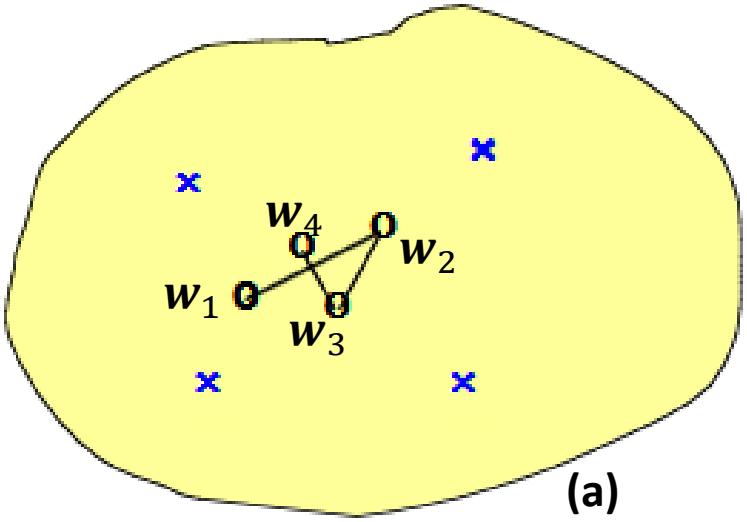
  - (D) *Parameter updating*

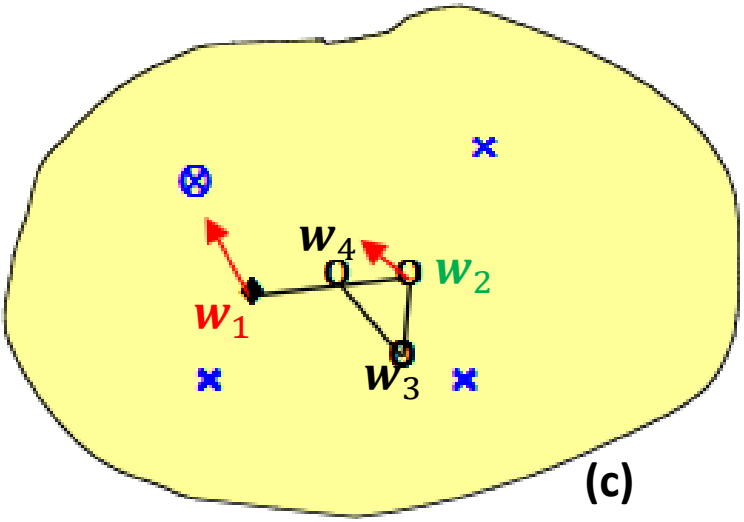
$$\boldsymbol{w}_k(t) = \begin{cases} \boldsymbol{w}_k(t-1) + \eta_t{}^{k,j}\big(\boldsymbol{x} - \boldsymbol{w}_k(t-1)\big), & if\ \boldsymbol{w}_k \in Q_j(t) \\ \boldsymbol{w}_k(t-1), & otherwise \end{cases}$$

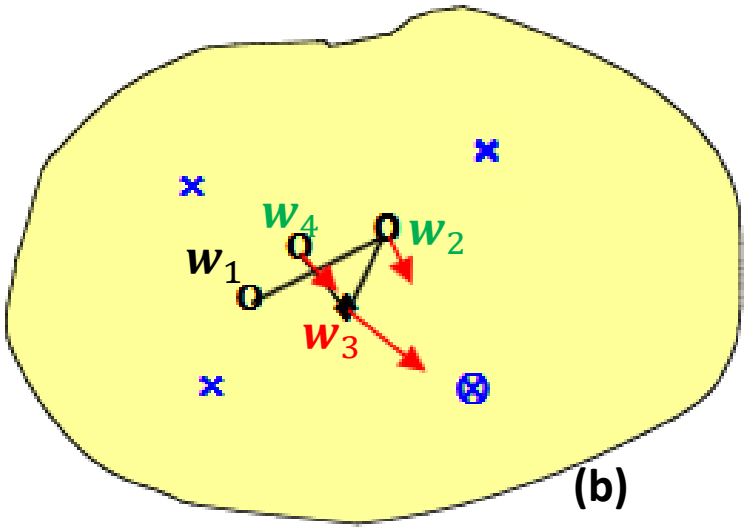
  - End
- (E) **Until** (convergence occurred) OR $(t > t_{max})$

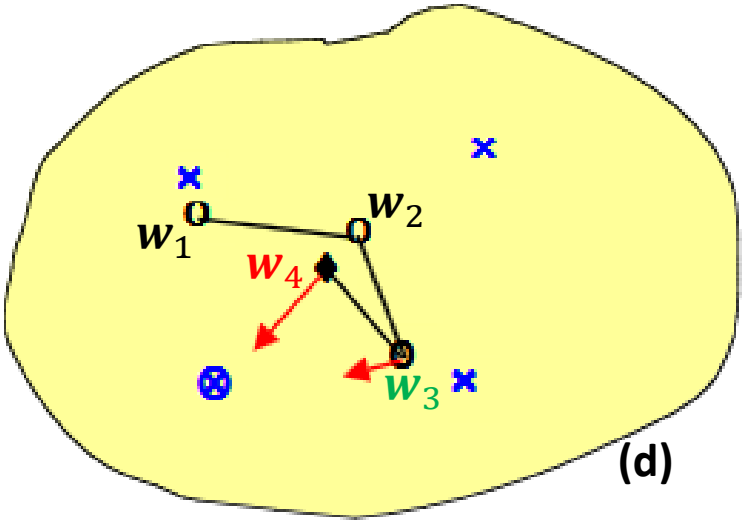
Self-organizing maps (*)

Example





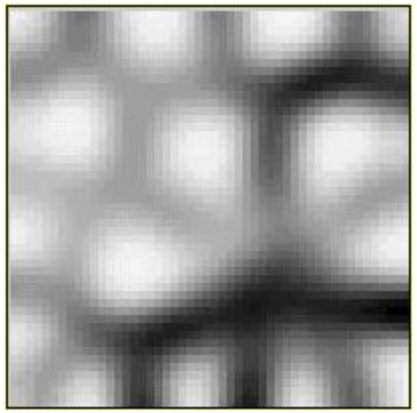




Self-organizing maps (*)

How to represent the result of a SOM

NOTE: After SOM convergence the topological ordering of the m **representatives** will comply with their "geometrical ordering".



Produce an **image** A of size

- m for the 1-d case or
- $k \times k$ for the 2-d case ($m = k^2$)

As follows

For **each representative** (pixel of A):

Compute its **average distance** d_{avg} from its neighboring representatives

Draw the associate pixel of A with a color so that:

The larger the d_{avg}, the darker the color will be.

Then **lighter areas surrounded** by **darker areas** in A are indicative of clustering structure in the data.

Supervised Learning Vector Quantization (VQ)

In this case

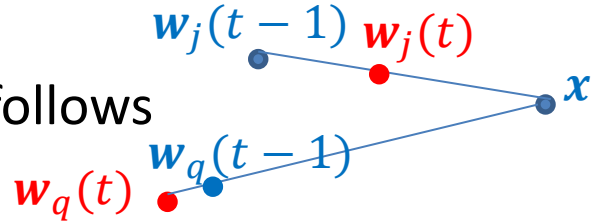
- each cluster is **treated** as a class (m compact classes are assumed)
- the available vectors have known class labels.

The goal:

Use a set of m representatives and place them in such a way so that each class is "optimally" represented.

The simplest version of VQ (LVQ1) may be obtained from GCLS as follows:

- Parts (A), (B) and (C) are the same with the basic competitive learning scheme.
- In part (D) the updating for $\boldsymbol{w}_j$' s is carried out as follows



$$\boldsymbol{w}_j(t) = \begin{cases} \boldsymbol{w}_j(t-1) + \eta(t)\left(\boldsymbol{x} - \boldsymbol{w}_j(t-1)\right), & if\ \boldsymbol{w}_j\ correctly\ wins\ on\ \boldsymbol{x} \\ \boldsymbol{w}_j(t-1) - \eta(t)\left(\boldsymbol{x} - \boldsymbol{w}_j(t-1)\right), & if\ \boldsymbol{w}_j wrongly\ wins\ on\ \boldsymbol{x} \\ \boldsymbol{w}_j(t-1), & otherwise \end{cases}$$

Supervised Learning Vector Quantization (VQ)

The algorithm

- $t = 0$
- **Repeat**
  - $t = t + 1$
  - **Present** a new randomly selected $\boldsymbol{x} \in X$ to the algorithm.
  - (B) **Determine** the winning representative $\boldsymbol{w}_j$ on $\boldsymbol{x}$ as the one for which

$$d\big(\boldsymbol{x}, \boldsymbol{w}_j(t-1)\big) = \min_{k=1,\dots,m} d(\boldsymbol{x}, \boldsymbol{w}_k(t-1))$$

  - (D) *Parameter updating*

$$\boldsymbol{w}_j(t) = \begin{cases} \boldsymbol{w}_j(t-1) + \eta(t)\big(\boldsymbol{x} - \boldsymbol{w}_j(t-1)\big), & if\ \boldsymbol{w}_j\ correctly\ wins\ on\ \boldsymbol{x} \\ \boldsymbol{w}_j(t-1) - \eta(t)\big(\boldsymbol{x} - \boldsymbol{w}_j(t-1)\big), & if\ \boldsymbol{w}_j wrongly\ wins\ on\ \boldsymbol{x} \\ \boldsymbol{w}_j(t-1), & otherwise \end{cases}$$

- (E) **Until** (convergence occurred) OR $(t > t_{max})$ (max allowable no of iter.)

In words:

- $\boldsymbol{w}_j$ is moved:
  - Towards $\boldsymbol{x}$ if $\boldsymbol{w}_j$ wins and $\boldsymbol{x}$ belongs to the j-th class.
  - Away from $\boldsymbol{x}$ if $\boldsymbol{w}_j$ wins and $\boldsymbol{x}$ does not belong to the j-th class.
- All other representatives remain unaltered.

Valley seeking clustering algorithms

Let $p(\boldsymbol{x})$ be the density function describing the distribution of the vectors in X.

- Clusters may be **viewed** as peaks of $p(\boldsymbol{x})$ **separated** by valleys.

Thus one may

- **Identify** these valleys and
- Try to **move** the borders of the clusters in these valleys.

A simple method in this spirit.

Preliminaries

- Let the distance $d(\boldsymbol{x}, \boldsymbol{y})$ be defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{y} - \boldsymbol{x})^T A(\boldsymbol{y} - \boldsymbol{x})$$

where A is a positive definite matrix

- Let the local region of $\boldsymbol{x}$, $V(\boldsymbol{x})$, be defined as

$$V(\boldsymbol{x}) = \{\boldsymbol{y} \in X - \{\boldsymbol{x}\}: d(\boldsymbol{x}, \boldsymbol{y}) \leq a\}$$

where a is a user-defined parameter

- k_j^i be the number of vectors of the j cluster that belong to $V(\boldsymbol{x}_i) - \{\boldsymbol{x}_i\}$.
- $c_i \in \{1, \ldots, m\}$ denote the cluster to which $\boldsymbol{x}_i$ will be assigned.

Valley seeking clustering algorithms

Valley-Seeking algorithm

- **Fix** a.
- **Fix** the number of clusters m.
- **Define** an initial clustering X.
- **Repeat**
  - For $i = 1$ to N
    - **Find** j: $k_j^i = \max_{q=1,\ldots,m} k_q^i$
    - **Set** $c_i = j$
  - End For
  - For $i = 1$ to N
    - **Assign** $\boldsymbol{x}_i$ **to** cluster C_{c_i}.
  - End For
- **Until** no reclustering of vectors occurs.

Valley seeking clustering algorithms

The algorithm

- **Centers** a window defined by $d(\boldsymbol{x}, \boldsymbol{y}) \leq a$ at $\boldsymbol{x}$ and **counts** the points from different clusters in it.
- **Assigns** $\boldsymbol{x}$ to the *cluster with the larger number of points* in the window (the cluster that **corresponds** to the highest local pdf).

In other words:

- The boundary is **moved away** from the "winning" cluster.

Remarks:

- The algorithm is sensitive to a. It is suggested to perform several runs, for different values of a.
- The algorithm is of a mode-seeking nature (*if more than enough clusters are initially appointed, some of them will become empty*).

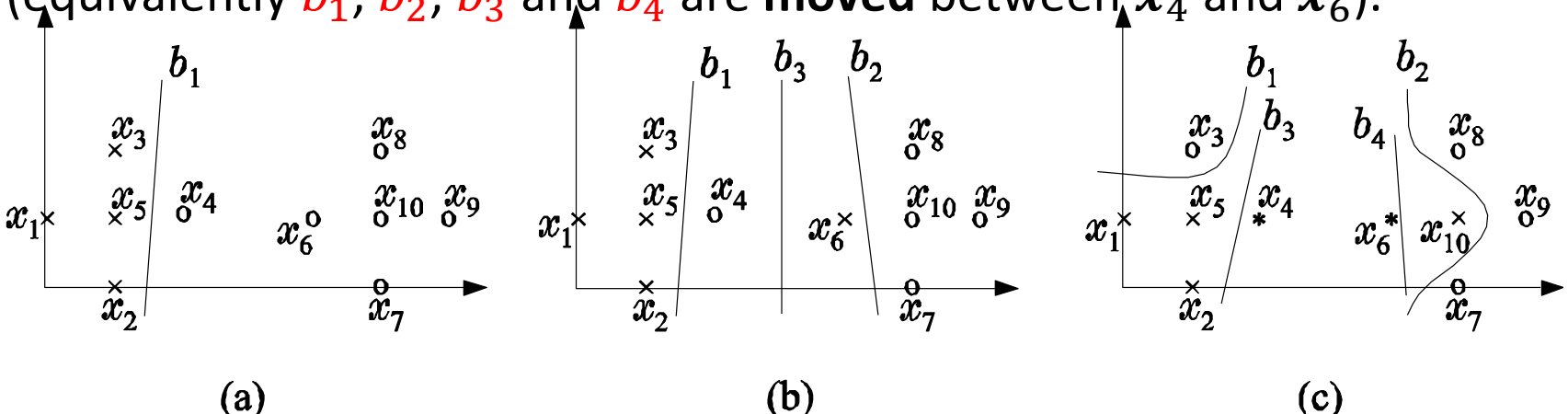
Valley seeking clustering algorithms

Example: Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{10}\}$ and $a = 1.1415\ (> \sqrt{2})$. X contains two physical clusters: $C_1 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5\}$, $C_2 = \{\boldsymbol{x}_6, \ldots, \boldsymbol{x}_{10}\}$.

(a) **Initially** two clusters are considered separated by b_1. After the convergence of the algorithm, C_1 and C_2 are identified (equivalently, b_1 is **moved** between $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$).

(b) **Initially** two clusters are considered separated by b_1, b_2 and b_3. After the convergence of the algorithm, C_1 and C_2 are identified (equivalently b_1 and b_2 are **moved** to the area where b_3 lies).

(c) **Initially** three clusters are considered separated by b_1, b_2, b_3, b_4. After the convergence of the algorithm, only two clusters are identified, C_1 and C_2 (equivalently b_1, b_2, b_3 and b_4 are **moved** between $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$).



Branch and Bound Clustering algorithms

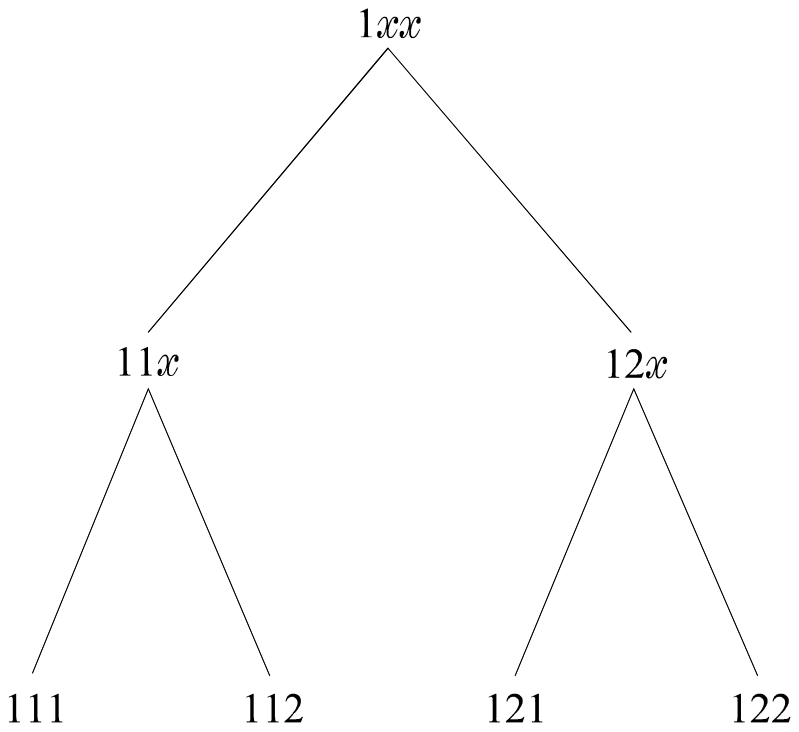
- They compute the globally optimal solution to combinatorial problems.
- They *avoid exhaustive search* via the employment of a monotonic criterion J.

Monotonic criterion J: *if* k *vectors of* X *have been assigned to clusters, the assignment of an extra vector to a cluster* ***does not decrease*** *the value of* J.

Consider the following 3-vectors, 2-class case:

121: 1st, 3rd vectors belong to class 1
2nd vector belongs to class 2.
(leaf of the tree)

$12x$: 1st vector belongs to class 1
2nd vector belongs to class 2
3rd vector is unassigned
(Partial clustering- node of the tree).



Branch and Bound Clustering algorithms

How exhaustive search is avoided

- Let B be the best value for criterion J computed so far.
- **If** at a node of the tree, the corresponding value of J is greater than B, **no further search** is **performed** for all subsequent descendants springing from this node.
- Let $\boldsymbol{C}_r = [c_1, \ldots, c_r]$, $1 \leq r \leq N$, denotes a partial clustering where $c_i \in \{1,2, \ldots, m\}$, $c_i = j$ if the vector $\boldsymbol{x}_i$ belongs to cluster C_j and $\boldsymbol{x}_{r+1}, \ldots, \boldsymbol{x}_N$ are yet unassigned.
- For compact clusters and fixed number of clusters, m, a suitable cost function is

$$J(\boldsymbol{C}_r) = \sum_{i=1}^{r} \left|\left|\boldsymbol{x}_i - \boldsymbol{m}_{c_i}(\boldsymbol{C}_r)\right|\right|^2$$

where $\boldsymbol{m}_{c_i}$ is the mean vector of the cluster C_{c_i}

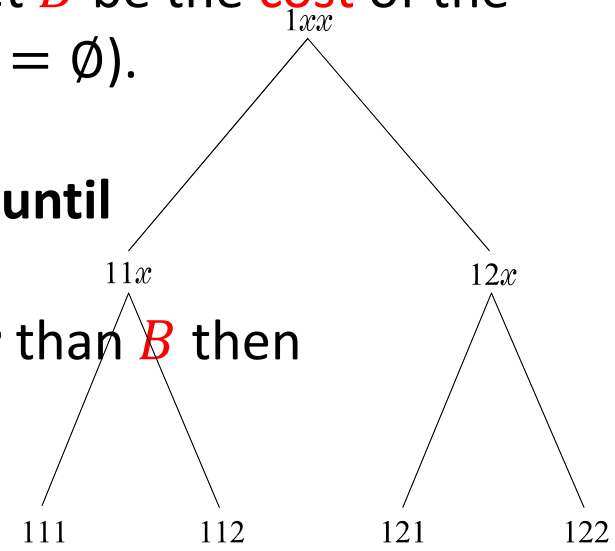
$$\boldsymbol{m}_j(\boldsymbol{C}_r) = \frac{1}{n_j(\boldsymbol{C}_r)} \sum_{\{q=1,\ldots,r, c_q=j\}} \boldsymbol{x}_q\,, \qquad j = 1, \ldots, m$$

with $n_j(\boldsymbol{C}_r)$ being the number of vectors $\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r\}$ that belong to cluster C_j.

Branch and Bound Clustering algorithms

Initialization

- **Start** from the initial node and go down to a leaf. Let B be the cost of the corresponding clustering $\boldsymbol{C}$ (initially set $B = +\infty$, $\boldsymbol{C} = \emptyset$).



Main stage

- **Start** from the initial node of the tree and go down **until**
  - **Either** (i) A leaf is encountered.
    - If the cost B' of the corr. clustering $\boldsymbol{C}'$ is **smaller** than B then
      - $B = B'$
      - $\boldsymbol{C} = \boldsymbol{C}'$ is the best clustering found so far
    - End if
  - **Or** (ii) a node q with value of J **greater** than B is encountered. Then
    - No subsequent clustering branching from q is considered.
    - **Backtrack** to the parent of q, q^{par}, in order to **span** a different path.
    - **If** all paths branching from q^{par} have been considered then
      - **Move** to the grandparent of q.
    - End if
  - **End if**

Terminate when all possible paths have been considered explicitly or implicitly.

Branch and Bound Clustering algorithms

Remarks

- Variations of the above algorithm, where much tighter bounds of B are used (that is, many more clusterings are rejected without explicit consideration) have also been proposed.

- A disadvantage of the algorithm is the excessive (and unpredictable) amount of required computational time.

Simulated Annealing

- It **guarantees** (under certain conditions) in probability, the determination of the globally optimal solution of the problem at hand via the minimization of a cost function J.
- It **may escape** from local minima since it **allows** moves that temporarily may increase the value of J.

Definitions

- An important parameter of the algorithm is the "temperature" T, which **starts** at a high value and **reduces** gradually.
- A sweep is the time the algorithm spends at a given temperature so that the system can enter the "thermal equilibrium" in this temperature.

Notation

- T_{max} is the initial value of the temperature T.
- $\boldsymbol{C}_{init}$ is the initial clustering.
- $\boldsymbol{C}$ is the current clustering.
- t is the current sweep.

Simulated Annealing

The algorithm:

- **Set** $T = T_{max}$ and $\boldsymbol{C} = \boldsymbol{C}_{init}$.
- $t = 0$
- **Repeat**
  - $t = t + 1$
  - *Repeat*
    - **Compute** $J(\boldsymbol{C})$
    - **Produce** a new clustering, $\boldsymbol{C}'$, by **assigning** a randomly chosen vector from X to a different cluster.
    - **Compute** $J(\boldsymbol{C}')$
    - **If** $\Delta J = J(\boldsymbol{C}') - J(\boldsymbol{C}) < 0$ then
      - (A) $\boldsymbol{C} = \boldsymbol{C}'$
    - Else
      - (B) $\boldsymbol{C} = \boldsymbol{C}'$, with probability $P(\Delta J) = e^{-\Delta J/T}$.
    - End if
  - *Until* an equilibrium state is **reached** at this temperature.
  - $T = f(T_{max}, t)$
- **Until** a predetermined value T_{min} for T is reached

Simulated Annealing

Remarks:

- For $T \to \infty$, it is $p(\Delta J) \approx 1$. Thus almost all movements of vectors between clusters are allowed.
- For lower values of T fewer moves of type (B) (from lower to higher cost clusterings) are allowed.
- As $T \to 0$ the probability of moves of type (B) tends to zero.
- Thus as T decreases, it becomes more probable to reach clusterings that correspond to lower values of J.
- Keeping T positive, we **ensure** a nonzero probability for escaping from a local minimum.
- We assume that the equilibrium state has been reached

  "If for k successive random reassignments of vectors, $\boldsymbol{C}$ remains unchanged."
- A schedule for lowering T that guarantees convergence to the global minimum with probability 1, is

$$T = \frac{T_{max}}{\ln(1+t)}$$

- The method is computationally demanding.